UNIVERSIDAD NACIONAL DE LOJA

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y LOS RECURSOS NATURALES NO RENOVABLES

Carrea de Sistemas Computacionales

AUTOR:

Ricardo Mathias Ochoa Armijos

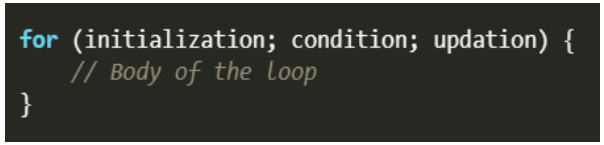
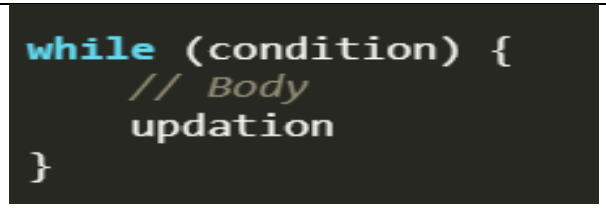
ING. LOPEZ FAICAN LISSETTE GEOCONDA

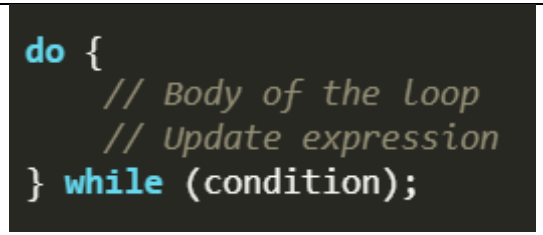
Teoría de la Programación

Cuadro comparativo entre las estructuras repetitivas

Loja Ecuador

2025

Tipo	Estructura	Uso	Características	Diferencias	Aplicaciones
FOR: El For en C es un tipo de estructura repetitiva o bucle**.**		El uso del for en C consiste en repetir un bloque de código un número conocido de veces. Se usa cuando ya tienes claro cuántas iteraciones vas a necesitar o cuando estás recorriendo algo que tiene límites definidos, como un arreglo o un rango de números.	Sirve para repetir algo un número claro de veces. Tiene inicialización, condición y actualización en una sola línea. La condición se revisa antes de cada vuelta. Es ordenado, fácil de leer y controlar. La variable del ciclo suele ser local al bucle.	La estructura está completa en una sola línea Se usa cuando sabes cuántas veces vas a repetir La condición se evalúa antes de cada repetición La variable del ciclo suele ser temporal y de control Ofrece más control visual del ciclo	Recorrer listas o arreglos Ejecutar tareas repetitivas un número exacto de veces Contar o generar secuencias numéricas Automatizar cálculos en bloque. Filtrar o buscar información Control de iteraciones en tareas dependientes
WHILE: El While en C es un tipo de estructura repetitiva**.**		El uso del while en n es repetir un bloque de código mientras una condición siga siendo verdadera. Es el bucle ideal cuando no sabes cuántas veces se repetirá la acción y dependes de algo que ocurre durante la ejecución.	No sabes cuántas veces se repetirá Depende totalmente de una condición lógica Requiere actualización interna.	La condición se evalúa antes de entrar No tiene límite fijo de repeticiones	Cuando no sabes cuántas veces se repetirá el proceso Esperar a que ocurra una condición externa

			Ideal para procesos que esperan algo Evalúa la condición antes de entrar Flexibilidad total	Se controla manualmente la actualización Es más flexible que un for	Validación de datos Recorrer procesos que dependen de estados cambiantes Repeticiones basadas en banderas lógicas Control de ciclos infinitos intencionales
DO WHILE: El do while es un bucle que primero ejecuta el bloque de código y luego evalúa la condición, garantizando que el ciclo se ejecute al menos una vez, sin excepción.		El uso del do while en C es repetir un bloque de código al menos una vez, sin importar la condición inicial, y seguir repitiéndolo mientras la condición siga siendo verdadera.	El código siempre se ejecuta al menos una vez Es un bucle de control por salida Ideal para procesos que requieren una primera ejecución obligatoria La condición determina si se repite o no.	La condición se evalúa al final, no al inicio El bloque se ejecuta mínimo una vez Es un bucle post-condicional. Ideal para interacciones donde siempre debe mostrarse algo antes. de preguntar.	Menús interactivos. Validación de datos Repetir procesos hasta que el usuario decida salir Simulaciones o cálculos que siempre arrancan al menos una vez Lectura de opciones o configuraciones iniciales

			La inicialización y la actualización no forman parte de la sintaxis Su estructura es compacta y fácil de leer Es útil cuando la lógica depende de interacción del usuario	La actualización no está integrada en la sintaxis.	Procesos que dependen del resultado anterior

Ejercicio:

Escribe un programa que dada las calificaciones de un grupo N con alumnos calcule el promedio del grupo. Entrada Un número n (1<= n <= 1000), representando el número de alumnos en la clase. Le siguen n líneas indicando las calificaciones de cada uno de los alumnos del grupo. Las calificaciones son números reales entre 0 y 10 con un sólo decimal.

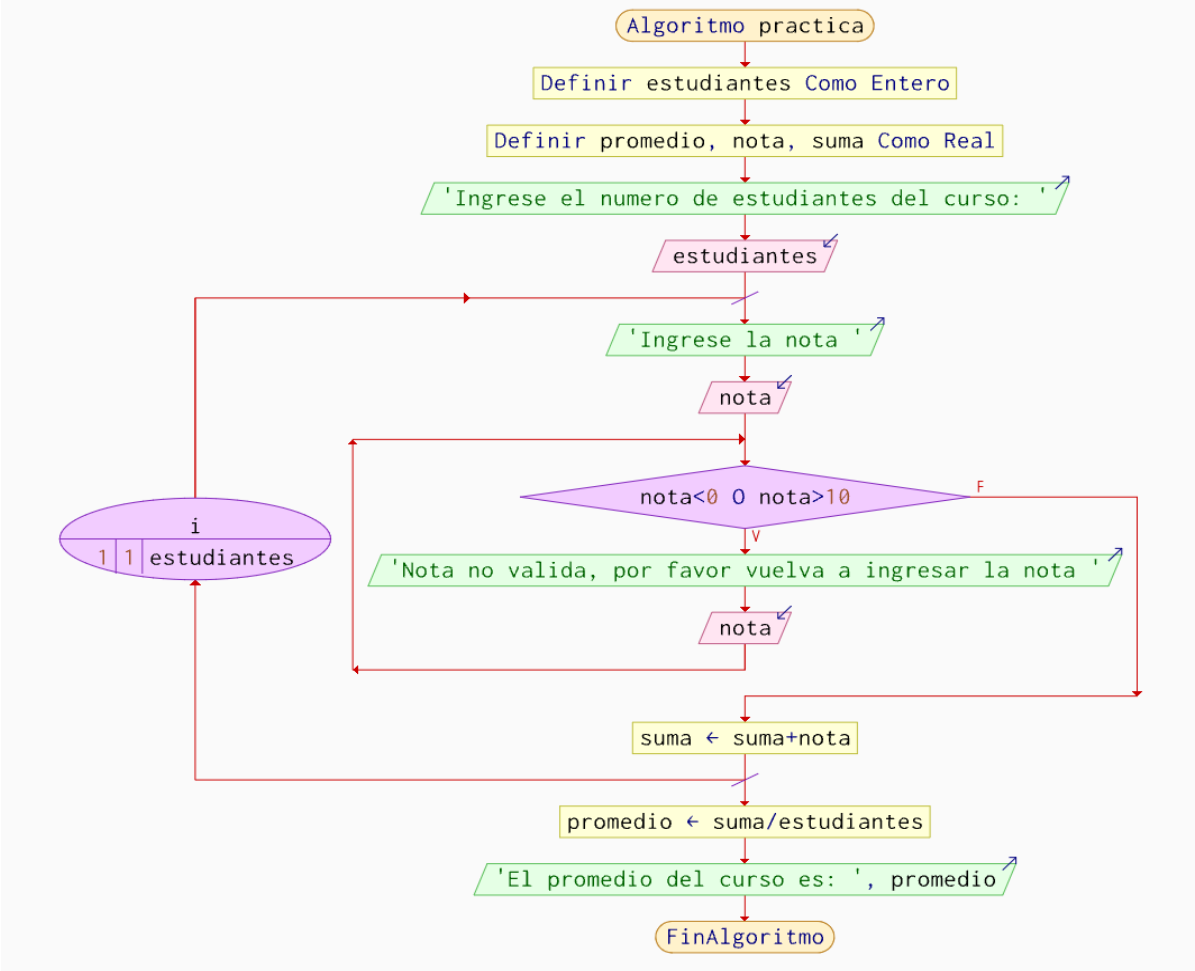
Salida

El promedio del grupo con una precisión de 2 decimales.

Ejemplo

Entrada	**Salida**
5 9.2 7.3 8.5 10.0 4.9	7.98

Diagrama de Flujo



Código en C

practica.c > main()

```c
#include <stdio.h>

int main(){

    int estudiantes;
    float promedio, nota, suma;

    printf("Ingrese el numero de estudiates del curso: ");
    scanf("%i", &estudiantes);

    for ( int i = 1; i <= estudiantes; i++)
    {
        printf("Ingrese la nota %i :", i);
        scanf("%f", &nota);

        while (nota < 0 || nota > 10)
        {
            printf("Nota no valida, por favor vuelva a ingresar la nota %i: ", i);
            scanf("%i", &nota);
        }
        suma += nota;
    }

    promedio = suma / estudiantes;

    printf("El promedio del curso es: %.2f", promedio);

    return 0;
}
```

Compilación

practica.c > ...

```c
#include <stdio.h>

int main(){

    int estudiantes;
    float promedio, nota, suma = 0;

    printf("Ingrese el numero de estudiates del curso: ");
    scanf("%i", &estudiantes);

    for ( int i = 1; i <= estudiantes; i++)
    {
        printf("Ingrese la nota %i :", i);
        scanf("%f", &nota);

        while (nota < 0 || nota > 10)
        {
            printf("Nota no valida, por favor vuelva a ingresar la nota %i: ", i);
            scanf("%f", &nota);  // ← Aquí está el arreglo
        }
        suma += nota;
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

```
PS C:\Users\USUARIO\Codigos Ricky\Unida 2> gcc practica.c -o practica.exe
PS C:\Users\USUARIO\Codigos Ricky\Unida 2> .\practica.exe
Ingrese el numero de estudiates del curso: 6
Ingrese la nota 1 :5.0
Ingrese la nota 2 :9.2
Ingrese la nota 3 :7.3
Ingrese la nota 4 :8.5
Ingrese la nota 5 :10
Ingrese la nota 6 :4.9
El promedio del curso es: 7.48
PS C:\Users\USUARIO\Codigos Ricky\Unida 2>
```

Conclusión

Las estructuras que son repetidas son esenciales para crear soluciones informáticas porque permiten la automatización de tareas que de otro modo serían largas, repetitivas y propensas a errores. Gracias a estos bucles, un programa puede procesar grandes cantidades de datos, comprobar información o realizar operaciones de forma continua sin obligar al programador a escribir las mismas instrucciones una y otra vez. Cada tipo de bucle se adapta a diferentes situaciones, brindando la flexibilidad para resolver problemas de manera efectiva. Una buena comprensión de ellos no sólo hace que el código sea más limpio y organizado, sino que también facilita el desarrollo de soluciones inteligentes que se ajusten exactamente a lo que necesitamos.

Referencias

[1] geeksforgeeks, «C while Statement,» geeksforgeeks, Noida, 2022.

[2] geeksforgeeks, «C do while ladder,» geeksforgeeks, Noida, 2022.

[3] geeksforgeeks, «C - for Statement,» geeksforgeeks, Noida, 2022.